



DEFENCE



DÉFENSE

DSSP20 (BETA EDITION)-USER GUIDE TO THE SIMULATION MODULES

M. Mackay

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC

Technical Memorandum

DREA TM 1999-109

August 1999



National
Defence

Défense
nationale

Canada



National Defence
Research and
Development Branch

Défense nationale
Bureau de recherche
et développement

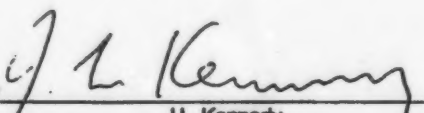
DREA TM 1999/109

DSSP20 (Beta Edition) - User Guide To the Simulation Modules

M. Mackay

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC
PO Box 1012, Dartmouth
Nova Scotia, Canada
B2Y 3Z7

Approved by


J.L. Kennedy
Head/Warship Signatures & Safety Section

August 1999

TECHNICAL MEMORANDUM

Prepared by

Defence
Research
Establishment
Atlantic



Centre de
Recherches pour la
Défense
Atlantique

Canada

Abstract

DSSP20 is a semiempirically-based suite of software modules for evaluating the dynamics and manoeuvring of streamlined underwater vehicles including submarines, smaller unmanned submersibles, and towfish. This memorandum is a user guide for the simulation modules which are used to estimate the forces and moments on a vehicle in a captive experiment (DSSP20B) and perform time domain simulations for a free-swimming vehicle (DSSP20C). It contains detailed descriptions of input preparation, brief discussions of output, and information for interpreting error messages.

Résumé

DSSP20 est un ensemble de modules logiciels à base semi-empirique servant à évaluer la dynamique et les manoeuvres de véhicules carénés sous-marins, notamment les sous-marins, les petits submersibles sans équipage et les corps remorqués. La présente note de service constitue un manuel de l'utilisateur pour les modules de simulation qui servent à estimer les forces et les moments exercés sur un véhicule dans une expérience captive (DSSP20B) et qui effectuent des simulations dans le domaine temporel pour un véhicule sous-marin libre (DSSP20C). Elle donne des descriptions détaillées de la préparation des données d'entrée, un bref aperçu des données de sortie ainsi que des renseignements sur l'interprétation des messages d'erreur.

DSSP20 (Beta Edition) – User Guide to the Simulation Modules

Michael Mackay

Executive Summary

Introduction

DREA Submarine Simulation Program, Version 2.0 (DSSP20), was developed to predict the manoeuvrability of submarines, AUVs, and towfish (the latter either in isolation or as part of a complete tow system). It generalizes the capability of the earlier Oberon-specific code DSSP02 that was used to study the tactical and safety implications of modifications to the submarine or to operating procedures — these investigations included towed array operation, ballast loss, sonar dome replacement, and diving limit redefinition. DSSP20 can model a wide range of vehicles and estimate their manoeuvring characteristics for a broad spectrum of system studies and evaluation, even preliminary design. It has been used for a systematic study of the effect of a hull plug (e.g., for AIP) on manoeuvring, and to support development of the CF Remote Minehunting System.

Significance

DSSP20 is still in development. However, the current beta edition of the code is being frozen and documented at this time since it is a useful tool which has already found application. The documentation will be updated as further progress with development is made.

Although input to DSSP20 input may initially appear complex because of a large number of options, defaults are provided that allow standard vehicle types (such as submarines) and manoeuvres to be defined with only a few parameters. This memorandum is the second in a series of user guides intended to facilitate the preparation of input, and addresses the requirements of the simulation modules. These use a vehicle description set up by the preprocessing modules — described in the first user guide — to simulate captive vehicle experiments and free-swimming vehicle manoeuvres. The guide is intended for both experienced and new users.

Future Development

Additional capabilities to be incorporated into the production edition of DSSP20 include provision for full autopilot control, completion of the time history algorithms, and calculation of compartment flooding, ballast tank blowing, and trim correction. In the longer term, a hybrid-calculation mode may be developed to take best advantage of available experimental data and CFD analysis.

Table of Contents

Page No.

	Abstract	ii
	Executive Summary	iii
	Table of Contents	iv
	Nomenclature	vi
1	Introduction	1
2	General Considerations	2
2.1	File Nomenclature	2
2.2	User Input	3
2.3	Axis Systems	4
2.4	Applicability	5
3	Control Deflections	5
3.1	Direct Deflection Input	6
3.2	Quasi-Direct Deflection Input	6
3.3	Indirect Deflection Input	7
3.4	Global Deflection Input	8
3.5	Conflicting Deflection Commands	8
3.6	Examples	9
4	DSSP20B: Single Vehicle Captive Simulation	9
4.1	Input and Output Files	9
4.2	Input File Specification	10
4.3	Level 1 Keywords	10
4.3.1	Run Type Keywords	11
4.3.2	Control Parameter Keywords	12
4.3.3	Text and Title	12
4.3.4	Switch Keywords	12
4.3.5	Other Parameter Keywords	13
4.4	Level 2 Static Experiment Keywords	13
4.4.1	Examples	13
4.5	Input Example	15
4.6	Interactive Dialogue	15
4.7	Output	16
5	DSSP20C: Single Vehicle Free Simulation	17
5.1	Input and Output Files	17
5.2	Input File Specification	18
5.3	Initialization Keyword Records	20
5.4	Command Keyword Records	21
5.5	Interactive Dialogue	21
5.6	Output	25
6	Concluding Remarks	

	Page No.
Appendix A: Parser Dictionary List	26
Appendix B: <i>Root.BI</i> and <i>Root.CI</i> Input Summaries	27
Appendix C: Error Messages	30
References	42

Nomenclature

$\overline{BG}$	submerged metacentric height
CG	vehicle center of gravity
k_ϕ	appendage anhedral factor
K, M, N	roll, pitch, and yaw moment
n	propulsor RPM
n_0	initial propulsor RPM
p, q, r	roll, pitch, and yaw rate (rotational state variables)
t	time
u, v, w	axial, lateral, and normal velocities (translational state variables)
U_0	initial speed
W_{ID}, W_{IH}, W_{IR}	weights for indirect depth, heading, and roll control
x, y, z	vehicle body axes
x_0, y_0, z_0	vehicle reference origin
x_0, y_0, z_0	earth-fixed axes
X, Y, Z	axial force, sideforce, and normal force
δ	geometric angle of deflection
$\delta_{LLim}, \delta_{ULim}$	lower and upper control deflection limits
δ_0	initial control deflection
ϵ	integration tolerance (DSSP20C)
ρ	water density
ϕ	appendage anhedral or dihedral angle
ϕ, θ, ψ	Euler (attitude) angles
ϕ_0, θ_0, ψ_0	vehicle reference axis directions

1 INTRODUCTION

DREA Submarine Simulation Program, Version 2.0 (DSSP20), is a suite of software modules for evaluating underwater vehicle hydrodynamics and manoeuvring. The key calculations of hydrodynamic forces and moments on the vehicle are done by component based semiempirical methods which are sufficiently accurate for preliminary design and routine manoeuvring analysis [1]. Development is continuing to further improve these methods and widen their scope of application. The code is currently in beta edition; a number of features are not fully implemented or are partially complete, including time history effects, flooding and ballast blowing models, and autopilot control. Nevertheless, the beta edition — specifically, Edition 981009 — is documented here because it is a useful tool in its own right and is a stable benchmark for future releases.

A flowchart of DSSP20 is sketched in figure 1; the flow of logic is predominantly from left to right. Preprocessing functions of DSSP20 consist of vehicle input checking and calculation of as many global quantities and coefficients as possible [2]. The tool kit comprises subroutines that calculate hydrodynamic forces and moments, and track the time history of wake trajectories and other quantities. There is no direct user interaction with the tool kit; client software sets up modelling and simulation problems and solves them by calling appropriate subroutines from tool kit and utility libraries. As the name implies, client software is conceptually an external entity and not truly part of DSSP20. Nevertheless, modules for single vehicle captive and free simulations have been implemented for the program, and they are documented in this memorandum.

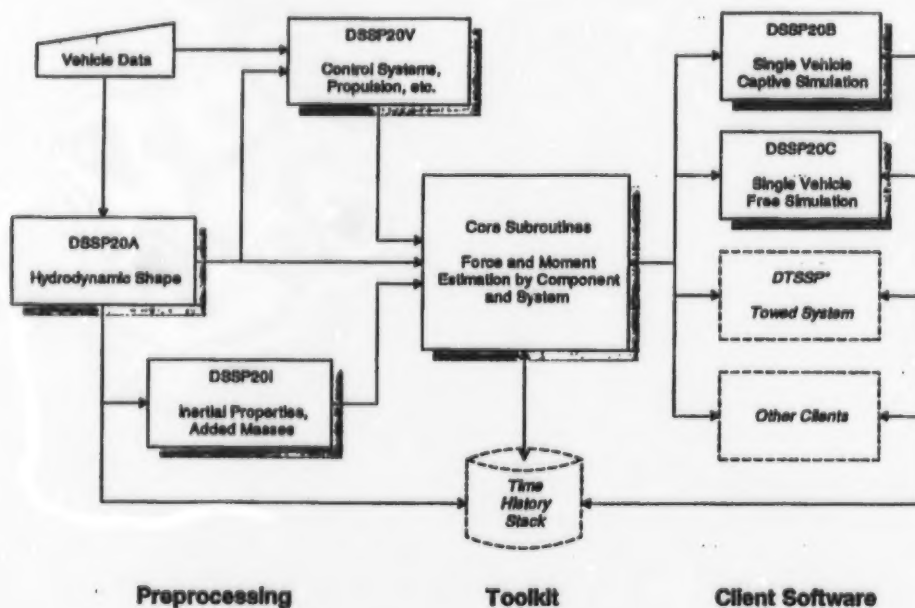


Fig 1 Flowchart of DSSP20. Items in dashed lines are projected or incomplete.

Each module of DSSP20 runs as a stand-alone program. For portability, the program user interface is minimal; most modules require a brief interactive dialogue using the keyboard and monitor. The code was written in FORTRAN 77 (with a couple of minor extensions) using the MS PowerStation Version 4.0 compiler for the PC. It has also been compiled and validated on a SGI Unix workstation.

As shown in figure 1, currently available client software is DSSP20B and DSSP20C:

DSSP20B calculates vehicle hydrodynamic forces and moments for a series of simulations in which a trajectory or values of the state variables are prescribed, i.e., as in a captive experiment. The inertial and added mass contributions are included; propulsor forces and moments are not. Control surface deflections are fixed – but may be different — for each simulation in the series.

DSSP20C does a free-swimming vehicle time simulation using order (3,4) Runge-Kutta integration with local error control. Propulsion is modeled.

The main objective of this user guide is to facilitate preparation of input files for DSSP20B and DSSP20C. The basic input requirements are quite simple, but a number of optional parameters are available to fine-tune the simulations. New users are advised to first read this guide completely, with particular attention to section 2, General Considerations, and to the numerous examples throughout the text. As a reference, more experienced users will need only sections 3, 4, and 5 (guides to control deflections, DSSP20B, and DSSP20C respectively), and the appendices, which contain additional information including input summaries and lists of error messages and their interpretation.

2 GENERAL CONSIDERATIONS

2.1 File Nomenclature

DSSP20 uses a standardized naming scheme for its input and output files. The filename is *Root.Ext* in which *Root* is a user-defined string up to eight characters long ("DSSP20" is used by default). For module DSSP20?, where '?' is 'B' or 'C' in the present context, file extensions *Ext* are interpreted as follows:

File	Function
<i>Root. ?I</i>	DSSP20? user-prepared input file
<i>Root. ?L</i>	DSSP20? run log output file
<i>Root. ?O</i>	DSSP20? output file to other DSSP20 modules
<i>Root. ?P</i>	DSSP20? printer output file

All output files are in ASCII characters and can be read (and manipulated) with any text editor. However, *Root. ?O* files, when read, are checked for consistency with the corresponding *Root. ?O* file [2], and a change made in them will result in a fatal error. If modification of the output is required, the module must be run again with appropriately modified input.

2.2 User Input

The input character set for DSSP20 comprises upper- and lower-case letters, the space, digits 0 to 9, and the "printable" characters `*()-+=;: ', <.>/'~!@#$%^&_ \{[]}'`. Tabs or other unrecognized characters in the input generally result in a fatal error.

In order to allow the user-prepared input files (*Root. ?I*) to be more readable, a simple input parsing scheme is employed. Each line in the input file is treated as a separate record of up to 71 characters consisting of words separated by one or more delimiters and terminated by an optional end-of-record character. Input is case-insensitive. The delimiters are the space ' ', comma ',', and semicolon ';'. The end-of-record (EOR) characters are the double quote '"', exclamation point '!', and percentage sign '%'. A word is a string of any other characters. Input following an EOR character is ignored and may therefore constitute a comment. Any input that is encountered after the input requirements of a record are fulfilled is similarly ignored.

Input is controlled by keywords; a keyword consists of an alphabetic string of up to ten characters contained in the parser's dictionary. A typical input record consists of a mixture of keywords and numerical arguments. Since some of these may be optional, it is prudent to always precede a following comment by an EOR character.

Keywords may be abbreviated by truncation on input so long as the result is not ambiguous, i.e., the abbreviation may not be the abbreviation of another word in the dictionary. For example, referring to the dictionary list in appendix A we see that 'Deb' is an acceptable abbreviation for 'Debug', whereas 'De' is not since the keywords 'Del', 'DelRateLim', and several other alternatives appear in the dictionary. On the other hand, an unabbreviated keyword takes priority over any abbreviated alternatives — 'Delta', 'U', and others are not considered to be ambiguous but are interpreted at face value. For this reason, keyword abbreviation should be used with care since the result may be unintended and inappropriate.

With keyword-driven input, records can in principle appear in any order in the input file. However, DSSP20B differs from other DSSP20 modules in that it processes the input file in sequence, and input order will generally affect the results. The structure of an input file may also be defined by input 'levels'. Input is initially at Level 1 and is switched to Level 2 and above by specific keywords. This concept is best illustrated by the DSSP20B input description given in sections 4.3 and 4.4.

Numerical values are also parsed. An integer input can have no more than 20 characters. It is read, right justified, in FORTRAN I20 format. A floating-point input can have no more than 30 characters. Initially it is read, right justified, in I30 format and converted to a double-precision variable; if that fails, it is re-read in F30.0 format into a double precision variable. The following are examples of valid input records defining the value of density, ρ :

```
Rho 1025.      ! Density in kg/m**3
RHO 1025       " Rho has only one argument, so ...
rho 1.025e3    no EOR character is required before the comment.
```

Text strings can be supplied for user comments on printed output; in general, they may contain any character and follow an appropriate keyword, e.g., Text or Title. A label string is used to uniquely identify a component. It consists of 1 to 24 alphabetic characters preceded by a label

character '#' or '?'. Component labels are input by the user in DSSP20A [2], and propulsor labels in DSSP20V [2]. They default to #HULL001, #LIFT001, #PROP001, etc.

Blank lines are ignored, and purely comment lines should start with an EOR character otherwise the parser will try to interpret the first delimited string as a keyword or numeric value. Appendix B gives input file summaries that illustrate some of the concepts discussed in this section.

2.3 Axis Systems

DSSP20 uses SI units; coordinates and lengths are in metres. There are two principal axis systems: vehicle axes, in which vehicle geometry and most other preprocessing input is defined, and reference axes, in which output quantities such as hydrodynamic forces and moments are defined. In addition, DSSP20C uses earth-fixed axes to output trajectory coordinates and vehicle Euler (attitude) angles. All axis systems are righthanded.

Vehicle axes are determined indirectly by the (x, y, z) coordinates of vehicle geometry which are input in module DSSP20A [2]. By default, reference axes are the same as vehicle axes, but they may be redefined by the user. To be most meaningful, the reference axes should coincide with conventional hydrodynamic axes [3], i.e., with the origin in the vicinity of the CG and with x directed forward, y to starboard, and z downward. Earth-fixed axes are coincident with the reference axes at the start of a DSSP20C simulation. Thus, the submarine's initial position is at the earth fixed origin, $(x_0, y_0, z_0)_{\text{start}} = (0, 0, 0)$.

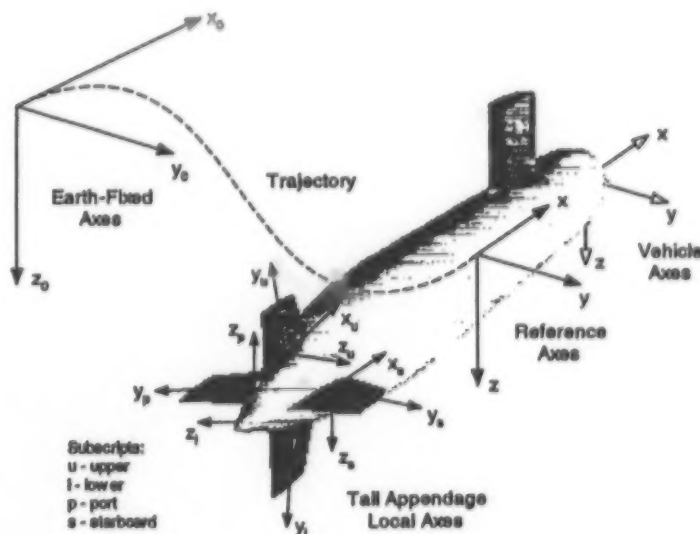


Fig 2 Example axis systems.

Internally, the program uses a separate local axis system for each component. For a lifting appendage the origin is at the root mid-chord, local y is outward along the mid-chord line, z is normal to the effective plane of the appendage, and x is obtained from their cross product. The principal reason for the user to be aware of local axes is that the deflection of flaps and all-moving appendages is defined in local axes: positive for rotation of z into x .

Examples of the various axis systems are sketched in figure 2. In this case, the vehicle axis origin is at the bow of the submarine. For all-moving rudders on the same shaft, the definition of appendage local axes results in deflections of opposite sign on the upper and lower rudders; the same observation applies to port and starboard sternplanes.

2.4 Applicability

It is implicit in many algorithms used in the current version of DSSP20 that the vehicle is streamlined and at moderate incidence, i.e., that vehicle components are reasonably slender and have their hydrodynamic axis — the longitudinal axis of a hull or the chordline of a lifting appendage — oriented more-or-less in the direction of motion (or against the direction of incident flow). The program therefore can not model bluff or irregular underwater vehicles such as ROVs, but can accommodate submarines, AUVs, and towfish. The geometry is currently described by up to two hull components and up to twelve lifting appendages; there can be up to three propulsors. Future versions will add component types for more diverse applications such as the semi-submersible Dolphin snort mast [4].

3 CONTROL DEFLECTIONS

DSSP20B and DSSP20C both use the same scheme and syntax for commanding control appendage deflections. The commands detailed in this section apply to both programs; minor differences in application are noted in sections 4.3.2, 5.3 and 5.4.

Control deflections are commanded by the keyword Delta or its synonym Del. The three principal modes of command can be summarized as:

- direct input, which allows the user to specify an individual appendage to be deflected,
- quasi-direct input, which indicates the type(s) of appendages and implied sense of resultant force direction for their deflection, and
- indirect input, which indicates the response of the vehicle and equivalent magnitude of the deflection.

There are also a few global control input commands.

For simulating captive model experiments with DSSP20B, this scheme has great flexibility at the expense of some complexity. However, the different modes of command are largely driven by the requirements for simulating free models, where quasi-direct or indirect commands are more generally used. In particular, autopilot control, which will be implemented in the production edition of DSSP20C, requires internal indirect commands which follow the rules and constraints outlined below.

Control deflection values (DSSP20B), or commanded values (DSSP20C), remain set until they are changed by another Delta command. Indirect input limits the changes that can then be made; it sets flags that allow only continued indirect input to the affected appendages until cancelled by the global Clear command (section 3.4). This has its origin in autopilot control for DSSP20C: a direct or quasi-direct command will not be permitted to "kick-out" an autopilot. In general, multiple deflection commands to the same appendage are not permitted (between runs for DSSP20B or simultaneously for DSSP20C); however, some combinations of multiple indirect commands are allowed — command conflicts are discussed in section 3.5.

3.1 Direct Deflection Input

The command is directed at a specific appendage:

Delta *Label String* , δ

Delta NL , δ

A control deflection of δ degrees in local coordinates is applied to the lifting appendage identified by the label string or by the integer NL. Attention must be given to sign because of the local coordinate convention (section 2.3). For example, to deflect horizontal sternplanes to generate upward force, the starboard plane requires positive δ while the port plane requires negative δ .

3.2 Quasi-Direct Deflection Input

The command is applied to one or more appendage types which are individually associated with a direction of the resulting control force. The notation $\pm\delta$ used below indicates that the sign of an appendage deflection depends on that direction; its magnitude is $|\delta|$.

Delta *keyword* , δ

The keyword is one of:

Bowplane : appendages with type LBowplane and SBowplane are deflected by $\pm\delta$ to produce an upward force.

Foreplane : appendages with type LBowplane, SBowplane, and Sailplane are deflected by $\pm\delta$ to produce an upward force.

Rudder : appendages with type Rudder are deflected by $\pm\delta$ to produce a force to starboard.

Sailplane : appendages with type Sailplane are deflected by $\pm\delta$ to produce an upward force.

Sternplane : appendages with type Sternplane are deflected by $\pm\delta$ to produce an upward force.

In this list, the resultant force direction is indicated for positive δ ; it will be opposite for negative δ . (Appendage types are discussed in reference [2].)

Quasi-direct input is appropriate for a cruciform arrangement of the aft appendages, and for horizontal foreplanes. X-rudders and other non-cruciform arrangements should have the type Tail, which does not respond to quasi-direct deflections.

3.3 Indirect Deflection Input

Indirect commands affect all control appendages that can contribute to the vehicle response indicated by the keyword. The magnitude of the deflection is weighted individually for each appendage and takes account of its anedral (see W_{ID} , W_{IH} , W_{IR} , and k_ϕ below). The notation $\pm \dots \delta$ is similar to that in the previous section; it indicates a deflection of $|\dots \delta|$ with the appropriate sign for the appendage.

Delta keyword, δ

Where the keyword is one of:

Down : appendages with type L_{Bow}plane, S_{Bow}plane, and Sailplane are deflected by $\pm W_{ID} k_\phi \delta$ to produce a downward force, and appendages with type Sternplane and Tail are deflected by $\pm W_{ID} k_\phi \delta$ to produce an upward force.

Port : appendages with type Rudder and Tail are deflected by $\pm W_{IH} k_\phi \delta$ to produce a force to starboard.

RollNeg : appendages with type Tail are deflected by $\pm W_{IR} \delta$ to produce a negative roll moment (i.e., counterclockwise looking forward).

RollPos : appendages with type Tail are deflected by $\pm W_{IR} \delta$ to produce a positive roll moment (i.e., clockwise looking forward).

Stbd : as Starboard, below.

Starboard : appendages with type Rudder and Tail are deflected by $\pm W_{IH} k_\phi \delta$ to produce a force to port.

Up : appendages with type L_{Bow}plane, S_{Bow}plane, and Sailplane are deflected by $\pm W_{ID} k_\phi \delta$ to produce an upward force, and appendages with type Sternplane and Tail are deflected by $\pm W_{ID} k_\phi \delta$ to produce a downward force.

Stbd is a synonym for Starboard; making it a keyword allows the common abbreviation Stbd to be used without ambiguity. The vehicle response indicated is for a positive value of δ ; the response is opposite if δ is negative. Therefore, Up +10 produces the same result as Down -10. For Up, Down, Port, and Stbd the commanded deflection is multiplied by anedral factor $k_\phi = \cos \phi$, where ϕ is appendage anedral in the appropriate crossflow direction (horizontal or vertical). The weight values W_{ID} , W_{IH} , or W_{IR} (for depth, heading, and roll, respectively) are defined in preprocessing, see reference 2, section 5.4.

Indirect deflection commands are processed independently according to whether they are associated with depth control (Up and Down), heading control (Port and Stbd), or roll control (RollNeg and RollPos). Although multiple input of any one mode is not permitted between runs, mixed

indirect input is allowed. Thus, for X-rudders with type Tail, the following multiple input:

Del Stbd 10
Del Down 5

will result in the rudders being deflected so as to produce both responses simultaneously.

An appendage under indirect control requires the global Clear command (next section) before it can be given direct or quasi-direct deflection commands.

3.4 Global Deflection Input

Delta *keyword*

Where the keyword is one of:

Clear : all control surfaces are returned to their initial deflection and control flags are cleared. The primary use of this command is to cancel indirect input. It is somewhat heavy-handed in the beta edition since it is done for all appendages simultaneously — the Clear command will be refined in later editions. Unlike Reset, below, Clear can be combined with other control deflection commands; regardless of order, Clear is executed first, then the other commands.

Reset : all control surfaces are returned to their initial deflection. (In this edition, initial deflections are equal to zero, so Reset has the same effect as Zero; this will be revised in later versions.) Reset cannot be combined with any other control deflection commands.

Zero : all control surfaces are set at zero deflection. Zero cannot be combined with any other control deflection commands.

3.5 Conflicting Deflection Commands

DSSP20 does not permit deflection command conflicts. Between runs for DSSP20B, or simultaneously for DSSP20C, only the following combinations of deflection commands are allowed for any individual appendage:

- a direct command with Clear,
- a quasi-direct command with Clear,
- an indirect-depth, indirect-heading, and indirect-roll command with each other, and with Clear,
- Reset, or Zero.

A fatal error is generated for any other combination.

In both DSSP20B and DSSP20C, commanded deflections stay in effect until legitimately canceled or replaced by a new command. An appendage under indirect control requires the global Clear command before, or coincident with, returning it to direct or quasi-direct control. Failure to do so generates a fatal error.

3.6 Examples

Examples illustrating control deflection input are given in section 4.5 for DSSP20B, and in section 5.2 for DSSP20C.

4 DSSP20B: SINGLE VEHICLE CAPTIVE SIMULATION

DSSP20B calculates the hydrodynamic forces and moments on a vehicle undergoing prescribed motion, as in a captive model experiment. In a static run, single values or ranges for the state variables $(u, v, w, p, q, r)^T$ are given, each set of values corresponding to a static towing tank (or wind tunnel) or rotating arm experiment. A set of steady forces and moments are output for each set of inputs. A sequence of static runs may be used to generate incidence sweep data for estimating stability derivatives. Time history runs simulate a dynamic captive model experiment, e.g., with the MDTF [5]. A sequence of values for $(u, v, w, p, q, r)^T$ is required, representing state variable time histories for the prescribed motion. Time histories of the resulting unsteady forces and moments are output.

4.1 Input and Output Files

DSSP20B input and output files follow the nomenclature introduced in section 2.1:

File	Function
<i>Root.BI</i>	DSSP20B user-prepared input file
<i>Root.BT</i>	DSSP20B time history file — required for the TimeHistory option
<i>Root.AO</i>	DSSP20A output file — required input to DSSP20B
<i>Root.IO</i>	DSSP20I output file — required for the TimeHistory option
<i>Root.VO</i>	DSSP20V output file — optional input
<i>Root.BL</i>	DSSP20B log output file
<i>Root.BO</i>	DSSP20B output file to other DSSP20 modules
<i>Root.BP</i>	DSSP20B printer output file

If *Root.VO* is not found, data input in DSSP20V (specifically the weights for indirect control, W_{ID} , W_{IH} , and W_{IR} , see section 3.3 and reference [2]) are ignored, and a warning is issued.

4.2 Input File Specification

In *Root.BI*, Level 1 input provides control of key run parameters and generates the sequence of runs to be simulated. Some keywords require additional Level 2 input.

Unlike other user-prepared input files in DSSP20, *Root.BI* is processed entirely sequentially at Level 1. Thus, a parameter definition applies to all simulation runs that follow it in the input file until the parameter is redefined. *Root.BI* input is summarized in appendix B.

A few consequences of sequential input in *Root.BI* will be addressed in subsequent editions of the program. For example, redefining a parameter without an intervening run is not presently trapped as an error. Also, although the (only one allowed) Title record may be in any position in the input file, having it as other than the initial record is not particularly useful.

4.3 Level 1 Keywords

The following table summarizes the keyword records which are valid at Level 1 of DSSP20B input. Optional arguments are enclosed in brackets, [].

Keyword	Argument(s)	Function
Debug		switches on debug output
Del	<i>Label String</i> or NL or <i>keyword</i> , δ	control deflection command
Delta	<i>Label String</i> or NL or <i>keyword</i> , δ	control deflection command
DeltaLim	<i>Label String</i> or NL, $\delta ULim$, $\delta LLim$	define control deflection limits
NoDebug		switches off debug output
Reference	[<i>Label String</i> or <i>keyword</i>]	define reference for angular motion and moments
Rho	ρ	define fluid density
Static	[u, v, w, p, q, r]	a static run
Text	<i>Text String</i>	descriptive text for printer output; ≤ 65 characters
TimeHistory	[<i>filename.ext</i>]	a time history run
Title	<i>Title String</i>	title for printer and other output; ≤ 64 characters

Keyword records are discussed below in terms of the functional group (i.e., run parameter, control parameter, and so on) that they belong to.

Unlike other DSSP20 program modules, DSSP20B executes Level 1 input in order, as it is read; this enables as many runs to be specified in the input file as required, and run types (Static, TimeHistory) can be freely mixed. To minimise input, parameter and control deflection settings are unchanged from run to run.

4.3.1 Run Type Keywords

The run type keywords are Static and TimeHistory; they define the captive vehicle experiment to be simulated.

Static [u, v, w, p, q, r]

Static forces and moments on the vehicle will be calculated for one or more combinations of state variable values. There are six collectively optional arguments which constitute the set of state variable values for a single static force and moment calculation. Either all, or none, of the arguments must be present. In their absence, input for a static experiment is input on subsequent records. This additional input is logically at Level 2, and is discussed in Section 4.4.

TimeHistory [filename.ext]

Total (dynamic and static) forces and moments will be calculated at each timestep of a time history file identified by the optional argument *filename.ext*. If this argument is absent, the default time history file *Root.BT* is assumed. A TimeHistory experiment also requires the file *Root.IO* (i.e., DSSP20I must be run first).

The time history file comprises a series of data records, one for each time step, each containing seven variables: t, u, v, w, p, q, r , where t is time. There must be at least three data records in the file. The usual rules of parsing apply (section 2.2), and data records may be interspersed with comments and blank records. Time can start at any positive or negative value, and must monotonically increase; time steps need not be equal. However, the steps should be small enough that a local quadratic fit to the data gives a reasonable estimate of the state variable time derivatives. The following example shows the start of a time history file.

```
| Example.BT - constant 0.1 Hz sampling rate, vertical plane motion
| t      u          v      w          p      q          r
-10.0  1.5          0      0.0          0      0.0          0
-9.9   1.500133715  0      0.0005814182563  0 -0.001052619113  0
-9.8   1.500502488  0      0.002136067918    0 -0.003969990336  0
-9.7   1.501054094  0      0.0043939595     0 -0.008406262175  0
```

4.3.2 Control Parameter Keywords

The control parameter keywords are Del, Delta, and DeltaLim. Del is a synonym for Delta; making it a keyword allows the common abbreviation Del to be used without ambiguity. Because *Root.BI* input is processed sequentially, control records affect only the runs that follow them in the input file.

Del *Label String or NL or keyword, δ*
As Delta, below.

Delta *Label String or NL or keyword, δ*
If the first argument parses as a label, it identifies the lifting surface appendage for which control deflection δ degrees is commanded. Or, if the argument can be converted to an integer, it is interpreted as the index NL of the selected appendage. These are direct input, section 3.1. Otherwise, the argument is parsed as a keyword for quasi-direct, indirect, or global input, sections 3.2, 3.3, and 3.4.

DeltaLim *Label String or NL, δ_{ULim} , δ_{LLim}*
An upper limit, δ_{ULim} degrees, and a lower limit, δ_{LLim} degrees, are defined for control surface deflections of the lifting appendage defined by the label string or integer index NL. δ_{ULim} and δ_{LLim} are real. The default limits are +45 degrees and -45 degrees respectively, set in DSSP20A.

4.3.3 Text and Title

Text and Title have arguments of ≤ 65 and ≤ 64 characters respectively and can be placed anywhere in the input file. There is no limit to the number of Text records. They are output to the printer output file *Root.BP*. There can be only one Title record; it is output to *Root.B0* as well as to printer output. In either case, the argument is not parsed, and may therefore contain the delimiters and EOR characters discussed in section 2; however, the normal end-of-record terminates the argument string.

4.3.4 Switch Keywords

The switch keywords in DSSP20B are Debug and NoDebug. They switch conditions within the program and, because *Root.BI* input is processed sequentially, affect only the runs that follow them in the input file.

Debug

Switches on debug output, sending values of many of the intermediate calculations to *Root.BL*. It should be used judiciously since a large volume of output may result.

NoDebug

Switches off debug output.

4.3.5 Other Parameter Keywords

The keywords which fall into this category are Reference and Rho. They override default values of parameters used by the program and, because *Root.BI* input is processed sequentially, affect only the runs that follow them in the input file.

Reference [Label String or keyword]

The reference is an origin and axes for defining the state variables and calculating forces and moments. By default, the reference is as set in DSSP20A. If the optional argument parses as a label, the reference is taken to be the local origin and local axes of the labelled component. Otherwise, the argument is parsed as a keyword. If the argument is the keyword CB then the reference origin is the calculated CB of the vehicle and the reference axes are vehicle axes. If neither of these conditions is met, or if the argument is absent, the reference origin and axes (angles of rotation from vehicle axes, in order, in degrees) must follow on the next two records as two triplets of real numbers:

$$\begin{matrix} x_o & y_o & z_o \\ \psi_o & \theta_o & \phi_o \end{matrix}$$

These data constitute a data block; that is, a blank record is not permitted between them, or between them and the Reference record.

Rho

ρ

This record overrides the default value of 1000.0 for fluid density, ρ . The argument is mandatory; its value is in kg/m^3 and must not be negative.

4.4 Level 2 Static Experiment Keywords

If the keyword *Static* is not followed by real arguments representing a single set of state variables, additional input is required to define ranges over which one or more of the state variables will be stepped. Each of these ranges is specified by a record of the form:

keyword , *lowerlimit* , *upperlimit* , *increment*

where

keyword is one of U, V, W, P, Q, or R corresponding to the state variables *u*, *v*, *w*, *p*, *q*, or *r* respectively.

lowerlimit is a real argument defining the lower range limit.

upperlimit is a real argument defining the upper range limit.

increment is a real argument defining the increment for the sequence of values.

State variables for which a range is not provided are assigned the value 0.0. Consequently, input for *u* must always be provided since DSSP20 requires that $u > 0$. All possible combinations of state variables are executed in the calculation; however, the maximum number of values currently allowed for any one state variable is 100 and the maximum total number of combinations is 1000.

4.4.1 Examples

Vary *w* and *q* over all combinations for fixed *u*:

```
Static      " For all combinations of w and q,
U 3.5 3.5 0.0 " fix u at 3.5 m/s,
W 0.0 0.5 0.1 " step w at .0 .1 .2 .3 .4 and .5 m/s, and
Q 0.0 0.2 0.05 " step q at .0 .05 .1 .15 and .2 rad/s.
```

Vary *w* and *q* independently for fixed *u*:

```
Static      " For fixed q=0, etc,
U 3.5 3.5 0.0 " fix u at 3.5 m/s,
W 0.0 0.5 0.1 " step w at .0 .1 .2 .3 .4 and .5 m/s,
Static      " then for fixed w=0, etc,
U 3.5 3.5 0.0 " fix u at 3.5 m/s,
Q 0.0 0.2 0.05 " step q at .0 .05 .1 .15 and .2 rad/s.
```

4.5 Input Example

The following DSSP20B input file exercises a number of the control deflection commands.

```
Title DELtaCOMmand tests " for type Tail cruciform appendages labelled #UpperRudder,
                        " #LowerRudder, #PortSternplane, and #StarboardSternplane
```

```

Static 1 0 0 0 0 0      " baseline static run; u = 1 m/s

Text Rudder torque test
Delta #Lower 5.0         " direct input (abbreviated labels)
Delta #Upper 5.0
Static 1 0 0 0 0 0

Text Rudder deflection test
Delta #Lower 5.0         " direct
Delta #Upper -5.0
Static 1 0 0 0 0 0
Del Stbd 20             " indirect-heading
Static 1 0 0 0 0 0

Text Plane deflection with rudders not canceled
Delta Up 10              " indirect-depth (no conflict)
Static 1 0 0 0 0 0

Text Now, plane deflection with rudders canceled
Delta Clear              " Clear cancels both indirect commands,
Delta Up 10              " so this one must be repeated
Static 1 0 0 0 0 0

Text ...and rudders with planes cancelled
Delta Port 20            " similar to previous; order of commands
Delta Clear              " does not matter (Clear is done first)
Static 1 0 0 0 0 0

Text Plane deflection test

Text RollPos RollNeg command tests
Delta RollPos 10         " superimposed on Port 20; this works because
Static 1 0 0 0 0 0       " appendages are type Tail, not Rudder and
Delta RollNeg 10         " Sternplane ...
Static 1 0 0 0 0 0
Delta RollNeg -10        " same as RollPos 10
Static 1 0 0 0 0 0

Text Test CLEAR keyword

Delta Clear              " baseline re-check
Static 1 0 0 0 0 0

Text Finally, run trajectory MEANDR

TimeHistory MEANDR.BT    " ... and run MEANDR

```

4.6 Interactive Dialogue

At the start of execution, the user is asked to supply the *Root* character string which is used to identify input and output files (see section 4.1):

Enter file name root (default = DSSP20):

A string up to eight characters long is expected; it should comply with the filename requirements of the host operating system. A carriage return indicates the default "DSSP20". If file *Root.A0* or *Root.BI* cannot be found, the following message is issued:

```
OPENIT failed to find or open [1]
- Enter new file name, including extension
```

where [1] is *Root.A0* or *Root.BI*. Note that entering the correct name for either of these files will not change the character string *Root*, and output files will still follow the naming convention of section 4.1. If either *Root.IO* or *Root.V0* are absent, a warning is issued.

In a TimeHistory run, absence of *Root.IO* generates a fatal error. Absence of the time history file (either as the argument to TimeHistory or the default, *Root.BT*) generates a warning and causes the run to be ignored.

4.7 Output

The output files from DSSP20B are:

Root.BL

This is the run log output file. It contains the input records prefixed by '>>', error messages (appendix C), and some internal run information. Its chief purpose is to aid error checking. The header and an example fragment from a typical example run log are:

```
DSSP20B log for filename root: dsub
...
?? Warning from SSP20B
   DSSP20.V0 not present - defaults assumed
-- DELCOM argument: Reset

Reading .BI input:

>>      Title DELtaCOMmand tests  " for type Tail cruciform appendages
>>                                     " #LowerRudder, #PortSternplane, and
>>
>>      Static 1 0 0 0 0 0          " baseline static run

-- Information from SSP20B
   The default values of XR will be used

-- Information from SSP20B
   The default value of RHO will be used
>>
```

```

>>      Text Rudder torque test
>>      Delta #Lower  5.0          " direct input (abbreviated 1
-- DELCOM argument:      Delta #Lower  5.0          " direct inp
>>      Delta #Upper  5.0
-- DELCOM argument:      Delta #Upper  5.0
>>      Static 1 0 0 0 0 0
>>
>>      Text Rudder deflection test

```

...

All Del and Delta commands are echoed to the log file when processed. The global Reset before reading *Root.BI* seen in the above example is an internal command that is issued as part of program initialization.

Root.B0

This is the output file for plotting and for future DSSP20 modules. Like all output, it is in ASCII characters. *Root.B0* currently uses a temporary format used for code development. Each run is preceded by blank records and column headers that have to be removed with a text editor for plotting. This format will be revised in subsequent editions of the program. Output from time history runs is not presently output to *Root.B0*.

Root.BP

This is the printer output file. Static runs output values of the state vector $(u, v, w, p, q, r)^T$ and corresponding force/moment vector $(X, Y, Z, K, M, N)^T$. Time history run output includes time derivatives of the state vector and dynamic and static contributions to the force/moment vector.

5 DSSP20C: SINGLE VEHICLE FREE SIMULATION

DSSP20C simulates a free-swimming vehicle that is manoeuvring in response to a sequence of control deflection and propulsor commands. The equations of motion are integrated using an order (3,4) Runge-Kutta method with adaptive timestep for local error control [6]. Output may be obtained at each timestep, or at a regular interval by means of interpolation [7].

Command input is not interactive. The input file is first sorted into initialization and command records; the latter into "bundles" of commands occurring simultaneously. Time integration from one command bundle to the next is called a "sprint". The simulation is terminated when the last sprint is complete; the Stop command is provided to define termination.

The vehicle is initially moving in a horizontal straight line, along the x_0 axis (section 2.3), in level trim and neutrally buoyant at depth $z_0 = 0$; initial speed is defined by the user. These conditions are maintained until a command is given which causes them to change.

DSSP20C is the most recent module for the program, and is consequently the least developed and mature. Capability to be added includes autopilot control, compartment flooding, ballast blowing, and trim correction. It is planned to allow initialization of more parameters (e.g., the reference axes) and to give the user more control over the integration process.

5.1 Input and Output Files

DSSP20C input and output files follow the nomenclature introduced in section 2.1:

File	Function
<i>Root.CI</i>	DSSP20C user-prepared input file
<i>Root.AO</i>	DSSP20A output file — required input to DSSP20C
<i>Root.IO</i>	DSSP20I output file — required input to DSSP20C
<i>Root.VO</i>	DSSP20V output file — required input to DSSP20C
<i>Root.CL</i>	DSSP20C log output file
<i>Root.CO</i>	DSSP20C output file to other DSSP20 modules
<i>Root.CP</i>	DSSP20C printer output file

5.2 Input File Specification

In *Root.CI*, input is predominantly at Level 1, where there are two types of input record, initialization records and command records. Initialization records are in the typical input form:

keyword , [*argument(s)*]

i.e., a keyword followed by one or more optional arguments that may themselves be keywords. These records are all executed before the time simulation is started, and are typically used to set up integration tolerance, initial speed, and other parameters. Initialization records are discussed in section 5.3.

Command records are distinguished by having a floating-point numerical value, *time*, as the first word in the record:

time , *keyword* , [*argument(s)*]

These records are executed at the indicated time during the simulation; they are typically commanded control deflections or changes in propulsor RPM. Command records are discussed in section 5.4.

The order of records in the input file is irrelevant. This facilitates input trial-and-error in simulating a manoeuvre since new commands being tested can be placed at the end of the input file where they can be easily modified without concern whether they are correctly ordered in sequence of time. However, once a manoeuvre is satisfactory, it is recommended that the input file be tidied up for archiving.

Two examples below illustrate flexibility in input; the first uses comments, formatting, and record grouping to achieve a reasonable degree of readability:

```
Title Horizontal ZigZag
!      10/10 zigzag at 10 kt, maintain depth +/- 2 m

0.      Start                ! Redundant in this run...
```

```

94.0 Stop | Stop at zero heading (got by trial and error)

Tolerance 0.001 | Redundant - same as default
UKT 10. | Initial speed 10 kt
Interpl 1 TypeOne | Interpolate with a one second interval
BG 0.3 | Define BG
RHO 1025. | Salt water

```

| ----- Commands:

```

0. Del Port 10 | 10 deg port rudder
7.5 Del Up 1.90 | Tweak for pitch-down
11.92 Del Stbd 10 | 10 deg starboard rudder - OVERSHOOT 1

14.0 Del Up 0.0 | Cancel pitch tweak
39.76 Del Port 10 | 10 deg port rudder - OVERSHOOT 2

56.0 Del Up 1.90 | Re-tweak pitch (depth)
63.0 Del Up 0.0 | Cancel pitch tweak
69.42 Del Stbd 10 | 10 deg starboard rudder - OVERSHOOT 3

```

The second example exaggerates poor input file preparation. It consists of the same input, but with redundancies, comments, and formatting removed, and with the records entered in random order. The numerical results from a simulation are identical, but the input file is now virtually undecipherable on quick inspection:

```

39.76 Del Port 10
Interpl 1 TypeOne
56.0 Del Up 1.90
11.92 Del Stbd 10
Title Horizontal ZigZag
Ukt 10
94.0 Stop
0. Del Port 10
Rho 1025.
63.0 Del Up 0.0
7.5 Del Up 1.90
Bg 0.3
69.42 Del Stbd 10
14.0 Del Up 0.0

```

5.3 Initialization Keyword Records

The following table summarizes the keywords that are valid for initialization records in DSSP20C input. Optional arguments are enclosed in brackets, [].

Keyword	Argument(s)	Function
BG	BG	define submerged metacentric height

Del	Label String or NL, δ_0	initialize control deflection
Delta	Label String or NL, δ_0	initialize control deflection
Interpl	Δt , [keyword]	define interpolation interval
Rho	ρ	define fluid density
RPM	[n_0]	initialize propulsor RPM(s)
Title	Title String	title for printer and other output; ≤ 64 characters
Tolerance	ϵ	define integration tolerance
Ukt	U_0 (kt)	initialize ahead speed, in kt
Ums	U_0 (m/s)	initialize ahead speed, in m/s

These records are interpreted as follows:

- BG** $\overline{BG}$
Define submerged metacentric height $\overline{BG}$ in metres. $\overline{BG}$ must not be negative. If this record is absent, a fatal error is generated if there is more than one Hull component; if there is only one Hull component, a default value of $D_M/20$, where D_M is maximum hull diameter, is used.
- Del** Label String or NL, δ_0 .
As Delta, below.
- Delta** Label String or NL, δ_0 .
An initial control deflection of δ_0 degrees in local coordinates is applied to the lifting appendage identified by the label string or by the integer NL. Attention must be given to sign because of the local coordinate convention (section 2.3). Non-zero δ_0 does not affect the vehicle initial condition of zero trim.
- Interpl** Δt , [keyword]
Define output interpolation interval Δt in seconds. Output in Root.CP and Root.CO is given for equal timesteps with this interval, although internal calculations are still done with the adaptive-timestep for error control. The optional keyword second argument indicates a fourth-order interpolant to use; it is TypeZero or TypeOne for, respectively, C^0 or C^1 continuity [6,7]. (The former, which is the default, is more economical but may perform less well in some circumstances.)
- Rho** ρ
This record overrides the default value of 1000.0 for fluid density, ρ . The argument is mandatory; its value is in kg/m^3 and must not be negative.
- RPM** [n_0]
Define initial value of RPM for all propulsors, and thus define, indirectly, initial speed. If there is only one propulsor, n_0 can be on the same record; otherwise, n_0 for each propulsor is given in a following data block, three to a record. There must be one, and only one, initialization record with the keyword RPM, Ukt, or Ums.
- Title** Title String
A title string of up to 64 characters is preceded by the keyword Title. The string is not parsed, and may therefore contain any character including delimiters and EOR characters. However, the normal end-of-record ("Enter" or "Carriage Return") terminates the string.

Tolerance ϵ

This record overrides the default value of 0.001 for integration tolerance. It must not be negative. Further work is required to generate recommendations for setting ϵ .

Ukt U_0 (kt)

Define initial speed in kt, and thus define, indirectly, propulsor RPM. There must be one, and only one, initialization record with the keyword RPM, Ukt, or Ums.

Ums U_0 (m/s)

Define initial speed in m/s, and thus define, indirectly, propulsor RPM. There must be one, and only one, initialization record with the keyword RPM, Ukt, or Ums.

5.4 Command Keyword Records

The following table summarizes the keywords that are valid for command records in DSSP20C input. Recall that the input format is:

time , keyword , [argument(s)]

Keyword	Argument(s)	Function
Del	<i>Label String</i> or NL or <i>keyword</i> , δ	control deflection command
Delta	<i>Label String</i> or NL or <i>keyword</i> , δ	control deflection command
Dummy		dummy command
Prop	[<i>Label String</i> or NL] , n	RPM command
Propulsor	[<i>Label String</i> or NL] , n	RPM command
Start		start simulation
Stop		stop simulation

These records are interpreted as follows:

Del *Label String* or NL or *keyword* , δ

As Delta, below.

Delta *Label String* or NL or *keyword* , δ

If the first argument parses as a label, it identifies the lifting surface appendage for which control deflection δ degrees is commanded. Or, if it can be converted to an integer, the argument is interpreted as the index NL of the selected appendage. These are direct input, section 3.1. Otherwise, the argument is parsed as a keyword for quasi-direct, indirect, or global input, see sections 3.2, 3.3, and 3.4.

Dummy

The Dummy command performs no function. However, it can be used as a time marker since it, like all command records, stops and restarts the integration.

Prop [*Label String* or NL] , n

As Propulsor, below.

Propulsor [Label String or NL], *n*

If the first argument is omitted, and there is only one propulsor, then that propulsor is commanded to *n* RPM; if there is more than one propulsor, a fatal error is generated. If the first argument is present, the propulsor identified by the label string or index NL is commanded to *n* RPM.

Start and

Stop

The Start and Stop commands are currently implemented the same way as Dummy; i.e., they perform no specific function. However, because of the way DSSP20C determines the bounds of the simulation — from the earliest command to the latest — they provide an easily recognisable way of defining the start and stop times. A warning is issued if the last command is not Stop since that command will not be executed. Additional functionality will be added to Start and Stop in future revisions of the program.

In this edition of DSSP20C, the Del or Delta keywords appear in both initialization and command records (other keywords will do so in future editions). The effect is different; in particular, the user should note that initialization is not the same as a command issued at the start of the simulation. Consider the following input:

```
tstart Delta #label 3
Delta #label -2
```

Input records can be in any order. In this example, the deflection of appendage #label is initialized to -2 degrees for the zero trim condition, then a change of deflection to +3 degrees is commanded at the start of the simulation.

5.5 Interactive Dialogue

At the start of execution, the user is asked to supply the *Root* character string which is used to identify input and output files (see section 5.1):

Enter file name root (default = DSSP20):

A string up to eight characters long is expected; it should comply with the filename requirements of the host operating system. A carriage return indicates the default "DSSP20". If file *Root.AO*, *Root.IO*, *Root.VO*, or *Root.CI* cannot be found, the following message is issued:

```
OPENIT failed to find or open [1]
- Enter new file name, including extension
```

where [1] is the expected filename. Note that entering the correct name for any of these files will not change the character string *Root*, and output files will still follow the naming convention of section 5.1.

5.6 Output

The output files from DSSP20C are:

Root.CL

This is the run log output file. It contains the input records prefixed by '>>', error messages (appendix C), and some internal run information. Its chief purpose is to aid error checking. The header and an example fragment from a typical example run log are:

DSSP20C log for filename root: dsubf

Looking for TAG on .AO input
Checksumming .AO file

...

Reading .CI input:

>>

>> Title DARPA SUBOFF F2_20

>> -40. Start

>> 120.0 Stop

>>

>> Tolerance 0.001

...

Initialization:

	Thrust	X-force	Cost
JS			
.950000	-50871.9043	-378091.3398	-428963.2441
1.000000	-84307.8415	-378091.3398	-462399.1812
.900000	-7232.9143	-378091.3398	-385324.2541
.850000	52229.5365	-378091.3398	-325861.8033
...			
.690601	378071.8511	-378091.3398	-19.4887
.690588	378107.9069	-378091.3398	16.5671
.690594	378089.8787	-378091.3398	-1.4611

DSSP20C started at -40.000000
DDNORK end step at -39.644344
DDNORK end step at -38.933032

...

DDNORK end step at -1.755204

SPRINT finished at .000000

>> PRPCOM argument: -242.554 "- Nominal RPM for 30 kt

DDNORK end step at .355656

...

DDNORK end step at 52.091394

DDNORK end step at 54.091394

!! Fatal Error in STATHL
UL is negative

Root.CO

This is the output file for plotting and for future DSSP20 modules. It contains time histories of the state vector, trajectory, and Euler angles: $t, u, v, w, p, q, r, x_0, y_0, z_0, \phi, \theta, \psi$.

Root.CP

This is the printer output file. It contains initialization information and time histories of the state vector, trajectory, Euler angles, control deflections, and propulsor RPM — but not in a format suitable for plotting. Some fragments of *Root.CP* corresponding to the *Root.CL* example follow:

```
--- DREA Submersible Simulation Program ----- Version 2.00 ---
--- DSSP20C - Free Model Simulation -----
----- (c) Copyright DND -----
```

Filename root: dsubf

...

=== Initialization: =====

...

=== Initial conditions for axial flight: =====

...

=== Legend: =====

t	- sec	u,v,w	- m/s
U	- kt	p,q,r	- deg/s
x0,y0,z0	- m	del	- deg
phi,the,psi	- deg	rpm	- rev/m

Labels:

del02	#LOWERRUDDER
del03	#UPPERRUDDER
del04	#PORTSTERNPLANE
del05	#STARBOARDSTERNPLANE
rpm01	#PROPO01

=== Time Stepping: =====

t =	-40.00	U=	20.00						
x0=	.0	phi=	.0	u=	10.29	p=	.00	del02=	.00 rpm01= 164.0
y0=	.0	the=	.0	v=	.00	q=	.00	del03=	.00
z0=	.0	psi=	.0	w=	.00	r=	.00	del04=	.00
								del05=	.00

At t = -40.000
START

t =	-39.64	U=	20.00						
x0=	3.7	phi=	.0	u=	10.29	p=	.00	del02=	.00 rpm01= 164.0
y0=	.0	the=	.0	v=	.00	q=	.00	del03=	.00
z0=	.0	psi=	.0	w=	.00	r=	.00	del04=	.00
								del05=	.00

• • •

01= 164.0

-242.554 "- Nominal RPM for 30 kt

01- 160.2

• • •

m01= -242.6

01= -242.6

pm01= -242.6

6 CONCLUDING REMARKS

This document is a user guide to the simulation modules of DSSP20, beta edition, with particular emphasis on input preparation. The guide is aimed at both experienced and new users. A user guide to the preprocessing modules has been published in a separate document.

The production edition of DSSP20 will contain a number of features that are incomplete in the beta edition. Validation of the semiempirical algorithms is on-going and improvements will continue to be made as new data become available. However, it is reasonable to expect that a model based on measured data, or possibly CFD analysis, will generally be more accurate for a specific vehicle. To take advantage of this, longer-term code development will explore the feasibility of providing a hybrid-calculation mode in which additional models can be readily introduced to supplement the semiempirical estimates.

APPENDIX A: Parser Dictionary List

The following 130 words are contained in the parser dictionary list, Edition 981009. Their internal representation is upper case, but user input is case-insensitive. The parser dictionary is used for input to all program modules and internally by the program.

ACONCOEFF	ACROSPIN	ADDEDMASS	ALL	ALLMOVING
BASEAREA	BG	BLOW	BOWPLANE	CAMBER
CB	CG	CLEAR	CXJ	CXZERO
DEBUG	DEL	DELRATELIM	DELTA	DELTADYN
DELTALIM	DELNONLIN	DIAMETER	DIAMRATIO	DOWN
DSSPTWO	DUMMY	ESAMVINT	FLAPGAP	FLAPPED
FLAPRODRAG	FOREPLANE	HARMONIC	HASENDP	HELP
HOERNER	HULL	HULLSEGMNT	INERTIAL	INITDELTAS
INTERPL	ISENDP	KILL	LABEL	LBOWPLANE
LEFT	LENGTH	LIFT	LIFTSEGMNT	LOCATION
LOD	LUMP	MAPLE	MIDHULL	NODEBUG
NOSE	OPEN	P	PITCH	PLANFORM
PLOT	POD	PORT	POWER	PROP
PROPULSOR	Q	R	RADIUS	REFERENCE
REFLECTED	RESET	RESIDRAGR	REVDYNAM	REVLIMIT
REVRATELIM	RHO	RIGHT	ROLL	ROLLNEG
ROLLPOS	ROTATED	ROUGHALLOW	RPM	RUDDER
SAIL	SAILPLANE	SAVE	SBOWPLANE	SHIPSTAB
SPREADSHT	STAMP	STARBOARD	START	STATIC
STATION	STBD	STERNPLANE	STOP	STRPROP
TAG	TAIL	TAILEFF	TEXT	THRUST
TIMEHISTRY	TITLE	TITLA	TITLB	TITLC
TITLI	TITLV	TOC	TOLERANCE	TORPEDO
TWIST	TYPEONE	TYPEZERO	U	UKT
UMS	UP	V	VEHICLE	W
WAGENINGB	WAKE	WEIGHTDCI	YAW	ZERO

APPENDIX B: Root.BI and Root.CI Input Summaries

The flow charts in this appendix summarise the input requirements for *Root.BI* and *Root.CI*, and illustrate how the input data streams are handled. The charts, on the following pages, are:

B.1 *Root.BI* input, and

B.2 *Root.CI* input.

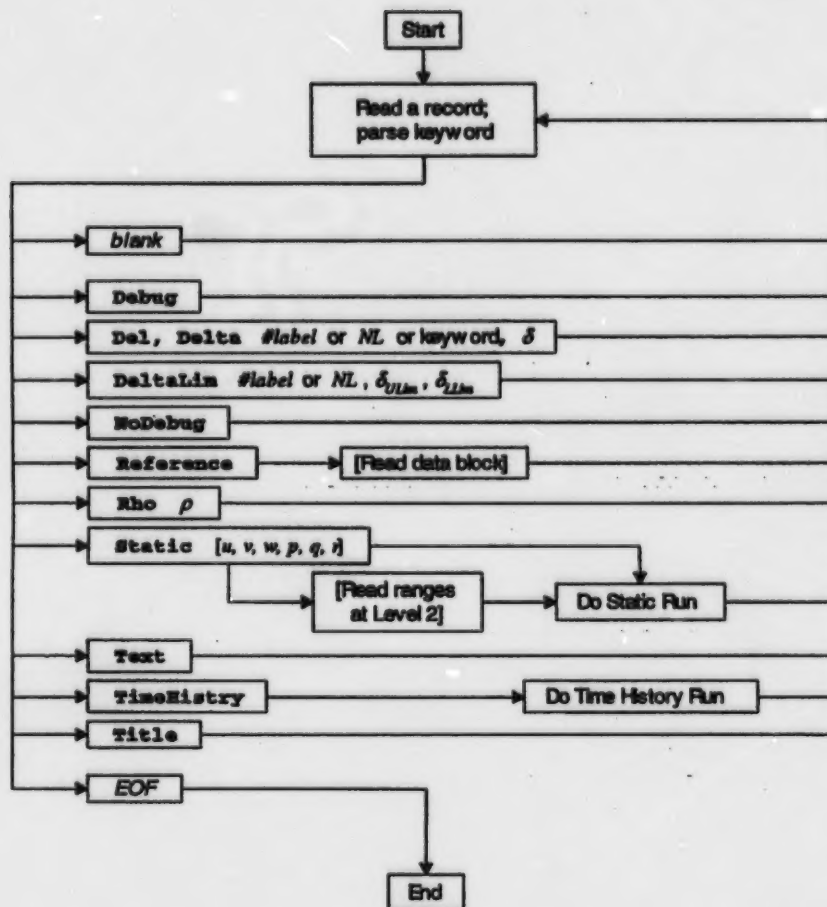


Fig B.1 Root.BI input.

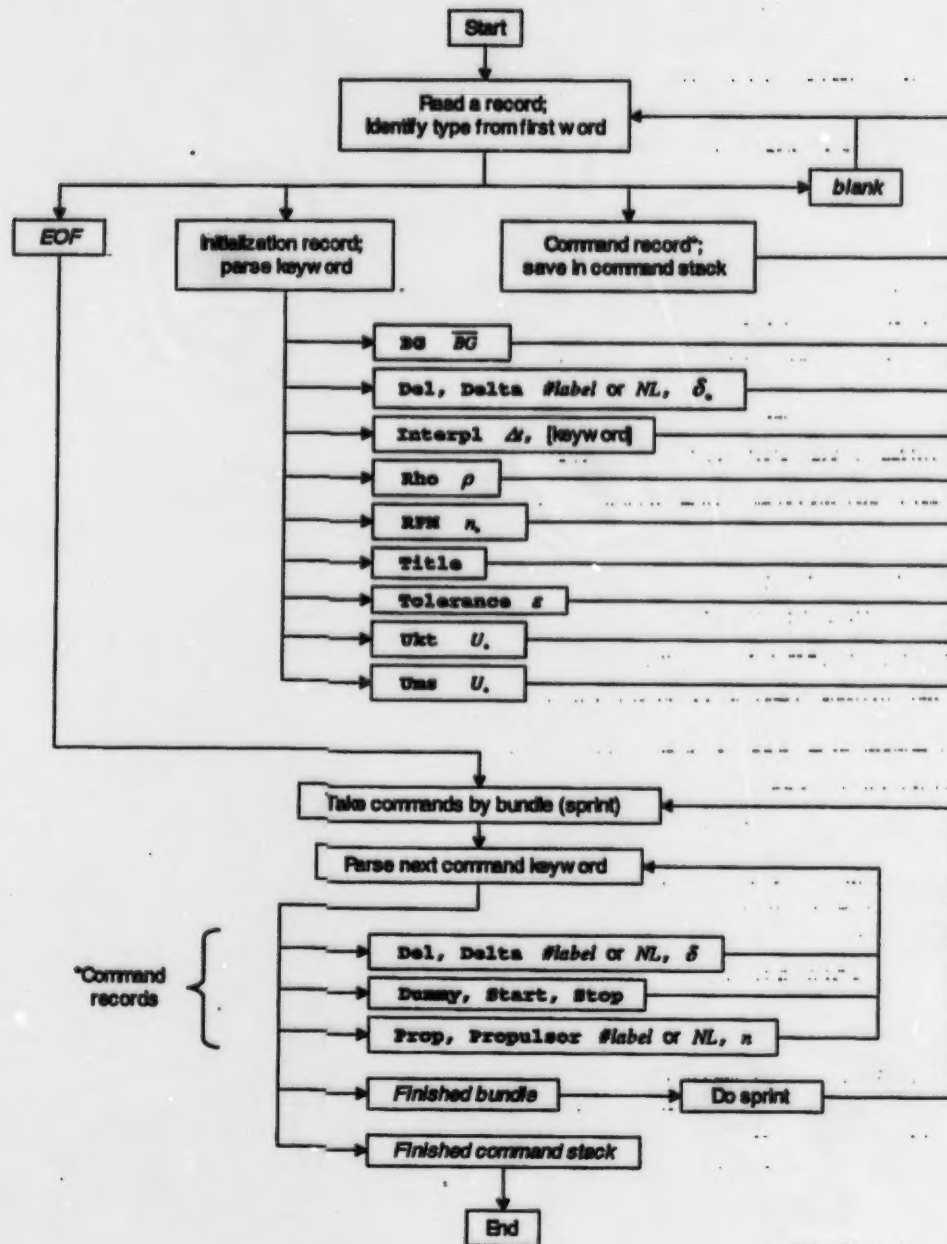


Fig B.2 Root.CI input.

APPENDIX C: Error Messages

This user guide is not, except in the broadest terms, concerned with the architecture of DSSP20. However, some familiarity with it is required because error messages cite the program unit (i.e., subroutine, function, etc.) in which the error occurred, thus providing a context for understanding the problem. Consequently, error messages are listed below by program unit and by the functional block, loosely corresponding to those on figure 1, in which the unit is located. The functional blocks pertaining to this document are:

- The main program and local subprograms for DSSP20B.
- The main program and local subprograms for DSSP20C.
- The DNORK integrator library used by DSSP20C.
- The core subprogram library, i.e., the tool kit.
- The utility subprogram library.

DSSP20 was developed piecemeal, using a large amount of error checking for local problems. As it was assembled, some error conditions became trapped in higher-level routines and the corresponding lower level error checks have become redundant. Errors that are believed to be redundant are indicated in the lists below with the notation "*impossible (?)*". A number of such instances may have been missed, and some may be misidentified. If a supposed redundant error is encountered, please inform the author, with the input file(s) and other supporting information, at mackay@dre.dnd.ca.

Some error messages are listed with the notation "*system*", indicating that they originate from compiler- or platform-related errors. These are typically 'ERR=' exits from a FORTRAN READ, WRITE, or OPEN command.

Error messages have four levels of severity: fatal, warning with pause, warning, and information. There may be one or more character strings or numerical values output as part of the message. In the lists below, these are indicated by [1], [2], and so on.

Fatal error messages are generated by a condition or problem that forces termination of program execution. They are issued to both the terminal and the module log file. The format is:

```
!! Fatal Error in program_unit_name
   message
```

A warning with pause is issued to indicate an actual or potential problem for which the user may wish to terminate program execution. These messages are issued to both the terminal and the module log file. The format is:

```
?? Warning from program_unit_name
   message
?? Continue execution ? ... Y/[N]
```

The default response is to terminate execution. The user's response is noted in the module log file as one of:

?! Execution terminated by user or
?? Execution continued by user

Warnings are issued to indicate an actual or potential problem while allowing program execution to continue. They are issued to both the terminal and the module log file. The format is:

?? Warning from *program_unit_name*
message

In a few instances, an information message is issued; this indicates that the program is flagging, at worst, a benign anomaly that the user may wish to consider. Information messages are issued to both the terminal and the module log file. The format is:

-- Information from *program_unit_name*
message

Many error messages are self-explanatory. This applies especially to those that arise from checking user input because input is echoed record-by-record to the module log file and the message will be displayed just after the offending input record. Typical examples from input to DSSP20B are:

!! Fatal Error in SSP20B
followed by:
Run title already found or
(RHO) value for rho missing or
(RHO) rho cannot be negative

The error lists in this appendix are sorted in alphabetical order of, first, the program unit in which the error was flagged and, second, the error message. The code 'F', 'P', 'W', or 'I' denotes, respectively, a fatal error, a warning with pause, a warning, or an information message. The lists do not include self-explanatory messages.

C.1 DSSP20B

This functional block comprises the following program units:

SSP20B Main program.
STATIC Coordinate static hydrodynamic calculations.
TIMHST Coordinate time history hydrodynamic calculations.
TIMEXN Calculate and output time history forces and moments.
TIQUAD Calculate vectors of quadratic coefficients from data in a ring array.

The error messages from this block that may require some clarification (given in italics) are:

W SSP20B DSSP20.IO not present
 — *required for TimeHistory runs only, see section 4.1*
W SSP20B DSSP20.VO not present - defaults assumed
 — *data input in DSSP20V will be ignored, see section 4.1*
F SSP20B Error reading .BI input
 — *system*

- F SSP20B Failed to open .AO input file
— cannot find Root.AO
- F SSP20B Failed to open input file
— cannot find Root.BI
- F SSP20B Failed to open output file
— system
- F SSP20B Failed to open printer file
— system
- F SSP20B Input .IO file not available; run DSSP20I
— Root.IO is required for TimeHistory runs
- F SSP20B Invalid keyword for .BI input: [1]
— keyword [1] is invalid for Root.BI input
- F SSP20B Invalid or ambiguous keyword: [1]
— input [1] does not match a dictionary keyword
- F SSP20B (REFERENCE) origin data missing
— first part of reference data block missing
- F SSP20B (REFERENCE) rotation data missing
— second part of reference data block missing
- F STATIC Argument(s) missing from state variable range
— arguments missing from Level 2 static input, see section 4.4
- F STATIC Argument(s) missing from STATIC statement
— all six arguments are required for Level 1 static input
- F STATIC Error reading .BI input
— system
- F STATIC Invalid or ambiguous keyword: [1]
— input [1] does not match a dictionary keyword
- F STATIC Multiple [1] ranges for Static run
— repeated range definition for [1] at Level 2
- F STATIC Too many state variable combinations
— limit on total number of combinations exceeded, see section 4.4
- F STATIC Too many values in range of [1]
— limit on total number of values for [1] exceeded, see section 4.4
- P STATIC Upper limit of [1] unmatched
— range of [1] is not an integer multiple of the increment
- F TIMHST Cannot extract root from ROOT !?
— impossible (?)
- F TIMHST Error reading ROOT.BT at [1]
— system
- W TIMHST TimeHistory abort - failed to open time history file
— Root.BT is missing and the TimeHistory run will be omitted
- F TIQUAD Incorrect stepping in X array
— time column in Root.BT is not monotonic

C.2 DSSP20C

This functional block comprises the following program units:

- SSP20C Main program.
- COMAND Execute command records.
- DYDT Define the extended equations of motion in canonical form, $\dot{y} = F(y, t)$.

SPCINI Set up printed output.
 SPCOUT Coordinate time history output.
 SPRINT Execute a sprint.

Integration and interpolation routines used by DSSP20C are in the DNORK integrator library, section C.3. Error messages from the DSSP20C functional block that may require some clarification (given in *italics*) are:

F SSP20C BG or CG undefined
 — *BG is undefined and there is more than one Hull (i.e., no default)*

F SSP20C CG is already set
 — *there is more than one BG record*

F SSP20C Command bundle size exceeded
 — *the bundle limit is 12 (simultaneous) commands*

P SSP20C Command count at tMax = [1]
 — *there is more than one command at the stopping time*

F SSP20C DELTA command argument invalid
 — *integer (control index) expected*

F SSP20C DELTA command on non-control appendage
 — *deflection command on appendage which is not Flapped or AllMoving*

F SSP20C DELTA index outside range
 — *invalid appendage index*

F SSP20C DELTA value outside range
 — *δ is outside default limits set in DSSP20A*

F SSP20C Error reading .CI input
 — *system*

F SSP20C Failed to open .AO input file
 — *cannot find Root.AO*

F SSP20C Failed to open command scratch file
 — *system*

F SSP20C Failed to open input file
 — *cannot find Root.BI*

F SSP20C Failed to open .IO input file
 — *cannot find Root.IO*

F SSP20C Failed to open output file
 — *system*

F SSP20C Failed to open printer file
 — *system*

F SSP20C Failed to open .VO input file
 — *cannot find Root.VO*

F SSP20C Initial speed or RPM already set
 — *there is more than one record with keyword RPM, Ukt, or Ums*

F SSP20C Initial U must be positive
 — *initialization record must give $U_0 > 0$*

F SSP20C (Interpl) Interpolation step not found
 — *Δt missing*

F SSP20C (Interpl) Invalid keyword: [1]
 — *keyword argument not TypeZero or TypeOne*

F SSP20C (Interpl) Invalid type: [1]
 — *expecting keyword for type (comment not preceded by EOR?)*

F SSP20C IntStp is already set
 — multiple Interpl records
 F SSP20C Invalid keyword: [1]
 — keyword [1] is invalid for DSSP20C input
 F SSP20C Invalid or ambiguous keyword: [1]
 — input [1] does not match a dictionary keyword
 F SSP20C Invalid simulation tMin,tMax - too few commands?
 — cannot determine simulation limits (boundaries) from commands
 F SSP20C Keyword missing from command record
 — second argument of presumed command record is not a keyword
 F SSP20C (RPM) No propulsors defined
 — impossible (?)
 F SSP20C No U or RPM initialization
 — one (and only one) RPM, Ukt, or Ums record is required
 P SSP20C Non-STOP command at tMax
 — stopping time command is not Stop
 F SSP20C RPM value missing (more than one PROP?)
 — missing data in RPM data block
 F CUMAND Invalid keyword: [1]
 — input [1] does not match a dictionary keyword
 F COMAND Keyword: [1] - out of context
 — keyword [1] is invalid for a command record
 F SPRINT Interpolation type error
 — impossible (?)
 F SPRINT Unexpected IND = [1] from DDNORK (interpolated output)
 — impossible (?)
 F SPRINT Unexpected IND = [1] from DDNORK (stepped output)
 — impossible (?)

C.3 DNORK Integrator Library

The integrator library comprises the following program units:

DDNORK Order (3,4) Runge-Kutta integrator.
 INTPC0 Fourth order interpolation, C^0 continuous.
 INTPC1 Fourth order interpolation, C^1 continuous.

There are only three error messages from this block. The integrator was implemented so that none should occur; they therefore have the notation "impossible (?)". Please report any instances to the author. The errors are:

F DDNORK IND out of range (1-6) on entry
 — impossible (?)
 F DDNORK Re-Entry at end of integration
 — impossible (?)
 F DDNORK TOL .LE. 0 on entry
 — impossible (?)

C.4 Tool Kit Library

The division of program units between the tool kit and utility libraries is somewhat arbitrary. The former contains units that are called by client software to perform the major functions of the tool kit, whereas the latter contains units that handle more general, bottom-level, functions for all parts of the program.

Both libraries consist of lower level routines than those in the functional blocks previously discussed, and more error messages from them are listed with the notation "*impossible (?)*" because errors are generally trapped, earlier, at the higher levels of client software written by the present author. However, these errors may occur, and be correctly reported, in the case of client software from other sources.

The tool kit library comprises the following program units:

CFTURB	Turbulent flow friction drag calculation.
CFWET	Hull friction drag estimation.
DELCOM	Interpret Del and Delta control records.
DELLIM	Reset control surface deflection limits.
DYNAMI	Dynamic inertial forces for complete vehicle.
DYNAMV	Dynamic added mass forces for complete vehicle.
KTKQ02	DSSP02-based propulsion model.
KTKQST	STR four-quadrant propulsion model.
KTKQWB	Wageningen B two-quadrant propulsion model.
MAIMAT	Calculate inertial and added-mass matrices.
NREFAO	Adjust <i>Root.AO</i> local axes to reference axes.
NREFVO	Adjust <i>Root.VO</i> local axes to reference axes.
PRPCOM	Interpret Prop and Propulsor records.
PRPFRG	Propulsor force and moment calculations.
PRPINT	Initialize propulsor conditions.
PRPINX	Calculate forces for initializing propulsor.
RATCOM	Rationalize multiple commands.
READAO	Read <i>Root.AO</i> and set up data.
READIO	Read <i>Root.IO</i> and set up data.
READVO	Read <i>Root.VO</i> and set up data.
STATHL	Estimate hull static forces and moments.
STATLT	Estimate appendage static forces and moments.

The error messages from this block that may require some clarification (given in italics) are:

F	CFWET	Error return from CFTURB; N = [1]
		— <i>impossible (?)</i>
W	CFWET	Flow transition region
		— <i>an approximation is used for hull drag</i>
W	CFWET	Laminar flow region
		— <i>an approximation is used for hull drag</i>
W	CFWET	Re below 1000
		— <i>an approximation is used for hull drag</i>
F	CFWET	Re cannot be negative
		— <i>impossible (?)</i>

F DELCOM Delta-command issued with RESET flag set
 — invalid combination of control commands
 F DELCOM Delta-command issued with ZERO flag set
 — invalid combination of control commands
 F DELCOM Invalid DELTA command argument
 — keyword argument not in dictionary
 F DELCOM LIFT([1]): Direct command flag preset
 — multiple direct commands to named appendage
 F DELCOM LIFT([1]): Indirect/Depth command flag preset
 — multiple indirect/depth commands to named appendage
 F DELCOM LIFT([1]): Indirect/Heading command flag preset
 — multiple indirect/heading commands to named appendage
 F DELCOM LIFT([1]): Indirect/Roll command flag preset
 — multiple indirect/roll commands to named appendage
 F DELCOM LIFT([1]): Quasi-direct command flag preset
 — multiple quasi-direct commands to named appendage
 F DELCOM RESET issued with command flag set
 — invalid combination of control commands
 F DELCOM ZERO issued with command flag set
 — invalid combination of control commands
 F DELLIM DELTALIM argument invalid
 — the argument must be a label or index
 F DELLIM Invalid delta limit range
 — lower limit $\geq$ upper limit
 F KTKQ02 Axial speed U must be positive
 — impossible (?)
 F KTKQ02 D(diameter) must be positive
 — impossible (?)
 F KTKQST D(diameter) must be positive
 — impossible (?)
 W KTKQST (Quad 1/2) negative J out of range
 — condition: $J_S < -1.85$
 W KTKQST (Quad 1/2) positive J out of range
 — condition: $6.15 < J_S$
 W KTKQST (Quad 3/4) negative J out of range
 — condition: $J_S < -6.15$
 W KTKQST (Quad 3/4) positive J out of range
 — condition: $2.05 < J_S$
 F KTKQWB Diameter must be positive
 — impossible (?)
 F PRPCOM (Case 2a,3a) Invalid reference to PROP [1]
 — propulsor index [1] is out of range
 W PRPCOM Integer argument converted to PROP 1 RPM
 — integer argument is interpreted as RPM for a single propulsor
 F PRPCOM Internal error: lost PROP argument
 — impossible (?)
 F PRPCOM Invalid first argument
 — keyword is not in dictionary
 F PRPCOM Invalid second argument
 — keyword is not in dictionary

F PRPFRG Propulsor index out of range
 — impossible (?)
 F PRPFRG UL is negative
 — impossible (?)
 F PRPFRG Unrecognised handedness : [1]
 — impossible (?)
 F PRPFRG Unrecognised propulsor type : [1]
 — impossible (?)
 F PRPINT Invalid flag INURPS : [1]
 — impossible (?)
 F PRPINT No propulsors defined
 — impossible (?)
 F PRPINT Non-convergence for bracketing JS
 — tolerance for convergence may be too small
 F PRPINT PROP [1] not a main propulsor
 — propulsor axis incidence greater than 20 deg relative to reference
 F RATCOM Clear and (Reset and/or Zero) flags set
 — inconsistent control commands
 F RATCOM Command conflict on LIFT([1]) - flags D and Q set
 — inconsistent control commands
 F RATCOM Command conflict on LIFT([1]) - flags I and (D and/or Q) set
 — inconsistent control commands
 W RATCOM Deflection limited on control surface [1]
 — requested deflection exceeds limit
 F READAO Error reading .AO input
 — system
 F READAO Hull component index missing from .AO
 — impossible (?)
 F READAO Hull component index on .AO out of order
 — impossible (?)
 F READAO Incomplete TAG on .AO file
 — impossible (?)
 F READAO Invalid or ambiguous keyword: [1]
 — impossible (?)
 F READAO Invalid STAMP on .AO input file
 — Root.AO has been modified
 F READAO Lifting component index missing from .AO
 — impossible (?)
 F READAO Lifting component index on .AO out of order
 — impossible (?)
 F READAO No TAG on .AO input file
 — impossible (?)
 F READAO Number of hull components on .AO exceeds MH
 — impossible (?)
 P READAO Title-A missing ...
 — impossible (?)
 F READIO Error reading .IO input
 — system
 F READIO Hull component data miscount
 — impossible (?)

F READIO Hull identification incorrect
 — impossible (?)
 F READIO Hull identification missing
 — impossible (?)
 F READIO Incomplete TAG on .IO file
 — impossible (?)
 F READIO Invalid STAMP on .IO input file
 — *Root.IO has been modified*
 F READIO Invalid or ambiguous keyword: [1]
 — impossible (?)
 F READIO Lift component data miscount
 — impossible (?)
 F READIO Lift identification incorrect
 — impossible (?)
 F READIO Lift identification missing
 — impossible (?)
 F READIO No TAG on .IO input file
 — impossible (?)
 F READIO TAGs on .AO and .IO files do not match
 — *Root.AO and Root.IO come from different input data*
 P READIO Title-I missing ...
 — impossible (?)
 F READVO AConCoeff unexpected for ML = 0
 — impossible (?)
 F READVO DelRateLim unexpected for ML = 0
 — impossible (?)
 F READVO Error reading .VO input
 — *system*
 F READVO Incomplete TAG on .VO file
 — impossible (?)
 F READVO Invalid STAMP on .VO input file
 — *Root.VO has been modified*
 F READVO Invalid or ambiguous keyword: [1]
 — impossible (?)
 F READVO No TAG on .VO input file
 — impossible (?)
 F READVO Number of propulsors on .VO exceeds MP
 — impossible (?)
 F READVO Propulsor index missing from .VO
 — impossible (?)
 F READVO Propulsor index on .VO out of order
 — impossible (?)
 F READVO TAGs on .AO and .VO files do not match
 — *Root.AO and Root.VO come from different input data*
 P READVO Title-V missing ...
 — impossible (?)
 F READVO WagingB unexpected for MP = 0
 — impossible (?)
 F READVO WeightDcI unexpected for ML = 0
 — impossible (?)

F STATHL COMP out of range
 — *impossible (?)*
 F STATHL UL is negative
 — *impossible (?)*
 F STATLT COMP out of range
 — *impossible (?)*
 F STATLT Delta out of nonlinear range !
 — *nonlinear flap estimate (DelNonLin) invalid for $\delta > 60$ deg.*

C.5 Utility Library

The utility library comprises the following program units:

ADDRNG Add vector to double precision ring array.
 ADIRNG Add vector to integer ring array.
 CHKDEC Convert from integer binary array.
 CHKINT Convert to integer binary array.
 CHKSUM Update running checksum.
 DECOMP Matrix decomposition for solving linear system.
 DEFLBL Initialize component label array.
 FCHECK Evaluate ASCII file checksum.
 GETCHR Get character variable from input stream.
 GETDBL Get double precision value from input stream.
 GETINT Get integer value from input stream.
 GDRNG Return vector from double precision ring array.
 GIRNG Return vector from integer ring array.
 INDICT Locate keyword in parser dictionary.
 INLABL Locate label in component label array.
 INTRPL Interpolation of single-valued function.
 MESSAG Error message manager.
 OPENIT Open file.
 QSFH Fourth-order quadrature.
 SOLVE Linear system solver.
 TOLABL Add label to component label array.
 UCCHAR Character case conversion and filtering.
 WORDIN Find delimited character string in input stream.

The error messages from this block that may require some clarification (given in italics) are:

F CHKSUM Character translation error for: [1]
 — *impossible (?)*
 F DECOMP Singular input matrix (small time step?)
 — *time step in a dynamic simulation may have become too small*
 F DEFLBL MZ exceeds I3.3 output format
 — *impossible (?)*
 F FCHECK Checksum found STAMP on NEW file
 — *impossible (?)*
 F FCHECK Incomplete STAMP on target file
 — *target file (Root.A0, etc.) may have been modified*

F FCHECK Invalid seed for checksum
 — impossible (?)

P FCHECK Invalid STAMP on target file
 — target file (Root.A0, etc.) has been modified

F FCHECK Invalid TYPE specification
 — impossible (?)

F FCHECK STAMP missing from target file
 — target file (Root.A0, etc.) may have been modified

F FCHECK Target file read error
 — system

F FCHECK Undetermined checksum error
 — impossible (?)

F GETCHR Non FORTRAN-extended character: may be a tab
 — see section 2.2

F INDICT Duplicate dictionary entries ?
 — impossible (?)

F INDICT Empty dictionary entry ?
 — impossible (?)

F INDICT Incorrect abbreviation switch: [1]
 — impossible (?)

F INLABL Duplicate component labels ?
 — impossible (?)

F INLABL Incorrect abbreviation switch: [1]
 — impossible (?)

F INLABL LABELS element [1] is empty
 — impossible (?)

F INLABL LABELS element [1] is null
 — impossible (?)

F INLABL LABELS element [1] missing label character
 — impossible (?)

F INLABL LLABWD out of range
 — a label string must be between 2 and 25 characters long including
 the initial label character

F INTRPL Data array not ascending in X
 — impossible (?)

F INTRPL Fewer than two data points
 — impossible (?)

F INTRPL Interpolation outside range
 — impossible (?)

F INTRPL Zero or negative interpolation points
 — impossible (?)

F MESSAG Incorrect MESTYP parameter .. MESSAG arguments are: [1] [2] [3]
 — (a multiline message) impossible (?)

F QSFM Dimension of input array < 3
 — impossible (?)

F TOLABL Filtered LABWRD is empty ?!!
 — impossible (?)

F TOLABL INDEX out of range
 — impossible (?)

W TOLABL Input word is a null label
 — word has only the initial label character

W TOLABL Input word is not a label
— *initial label character missing*

W TOLABL LABWRD duplicates label [1]
— *labels must be unique*

W TOLABL LABWRD is not alphanumeric
— *label contains non-alphanumeric characters*

F TOLABL LLABWD out of range
— *a label string must be between 2 and 25 characters long including
the initial label character*

F UCCHAR Incorrect filter option: [1]
— *impossible (?)*

REFERENCES

1. Mackay, M.: "Predicting the Effect of a Hull Plug on Submarine Manoeuvring using Semiempirical Methods", RINA International Symposium Warship '99, Naval Submarines 6, London, June 1999.
2. Mackay, M.: "DSSP20 (Beta Edition) — User Guide to the Preprocessing Modules", DREA Technical Memorandum 1999-108, August 1999.
3. Feldman, J.: "DTNSRDC Revised Standard Submarine Equations of Motion", DTNSRDC SPD-0393-09, June 1979.
4. Seto, M.L. and Watt, G.D.: "The Interaction Dynamics of a Semi-Submersible Towing a Large Towfish", Eighth International Offshore and Polar Engineering Conference, Montreal, May, 1998.
5. Mackay, M.: "Submarine Turning Circle Experiments in a Towing Tank", RINA International Symposium Warship '96, Naval Submarines 5, London, June 1996.
6. Enright, W.H., Jackson, K.R., Nørsett, S.P., and Thomsen, P.G.: "Interpolants for Runge-Kutta Formulas", ACM Transactions on Mathematical Software, Vol. 12, No. 3, September 1986.
7. Enright, W.H., Jackson, K.R., Nørsett, S.P., and Thomsen, P.G.: "Effective Solution of Discontinuous IVPs Using a Runge-Kutta Formula Pair with Interpolants", Applied Mathematics and Computation, Vol. 27, Elsevier, 1988.

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.) Defence Research Establishment Atlantic PO Box 1012 Dartmouth, Nova Scotia, Canada B2Y 3Z7	2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable). UNCLASSIFIED	
3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title). DSSP20 (Beta Edition) – User Guide to the Simulation Modules		
4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.) Mike Mackay		
5. DATE OF PUBLICATION (month and year of publication of document) August 1999	6a. NO. OF PAGES (total containing information include Annexes, Appendices, etc.) 42	6b. NO. OF REFS (total cited in document) 7
7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered). TECHNICAL MEMORANDUM		
8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address). Defence Research Establishment Atlantic		
9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant). Project 1gb14	9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written). 	
10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.) DREA TM 1999-109	10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor.) 	
11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification) (☒) Unlimited distribution () Defence departments and defence contractors; further distribution only as approved () Defence departments and Canadian defence contractors; further distribution only as approved () Government departments and agencies; further distribution only as approved () Defence departments; further distribution only as approved () Other (please specify):		
12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected).		

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

13. **ABSTRACT** (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

DSSP20 is a semiempirically-based suite of software modules for evaluating the dynamics and manoeuvring of streamlined underwater vehicles including submarines, smaller unmanned submersibles, and towfish. This memorandum is a user guide for the simulation modules which are used to estimate the forces and moments on a vehicle in a captive experiment (DSSP20B) and perform time domain simulations for a free-swimming vehicle (DSSP20C). It contains detailed descriptions of input preparation, brief discussions of output, and information for interpreting error messages

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

Underwater vehicles
Manoeuvring
Simulation
DSSP20

UNCLASSIFIED
SECURITY CLASSIFICATION OF FORM